1994-03

# A VLSI interface for the NM24CF04 serial-access ferroelectric memory

Dickerson, James H.

Monterey, California. Naval Postgraduate School

# NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A VLSI INTERFACE FOR THE
NM24CF04 SERIAL-ACCESS FERROELECTRIC
MEMORY

by

James H. Dickerson

March, 1994

Thesis Advisor: Douglas J. Fouts

# REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

| | | |
|---|---|---|
| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE 1994 March | 3. REPORT TYPE AND DATES COVERED Master's Thesis |

4. TITLE AND SUBTITLE A VLSI INTERFACE FOR THE NM24CF04 SERIAL- ACCESS FERROELECTRIC MEMORY

5. FUNDING NUMBERS

6. AUTHOR(S) James H. Dickerson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE A

13. ABSTRACT *(maximum 200 words)*

The goal of this research project was to design a VLSI implementation of the required digital circuitry to utilize ferroelectric memory as a portion of main microprocessor memory. An interface between National Semiconductor's NM24CF04, a nonvolatile, serial-access, ferroelectric memory device, and Intel's 8086 microprocessor was designed and implemented using SSI/MSI technology in a previous study. This thesis discusses the redesign of the previously designed circuit for VLSI implementation. The layout was accomplished using the Magic graphical layout editor and tested using the Esim event driven logic-level simulator.

14. SUBJECT TERMS Ferroelectric memory, FERRAM,NM24CF04,nonvolatile memory,VLSI

15. NUMBER OF PAGES 114

16. PRICE CODE

| | | | |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT Unclassified | 18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified | 19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified | 20. LIMITATION OF ABSTRACT UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

# A VLSI Interface for the NM24CF04 Serial-Access Ferroelectric Memory

by

James H. Dickerson
Lieutenant, United States Navy
B.S., University of Mississippi, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1994

Author: [redacted]
James H. Dickerson

Approved by: [redacted]
Douglas J. Fouts, Thesis Advisor

[redacted]
Chin-Hwa Lee, Second Reader

[redacted]
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

## ABSTRACT

The goal of this research project was to design a VLSI implementation of the required digital circuitry to utilize ferroelectric memory as a portion of main microprocessor memory An interface between National Semiconductor's NM24CF04, a nonvolatile, serial-access, ferroelectric memory device, and Intel's 8086 microprocessor was designed and implemented using SSI/MSI technology in a previous study. This thesis discusses the redesign of the previously designed circuit for VLSI implementation. The layout was accomplished using the Magic graphical layout editor and tested using the Esim event driven logic-level simulator

Thesis
L577

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# I. INTRODUCTION

## A. BACKGROUND

Ferroelectric capacitor memory (FECM) cells are incorporated in the National Semiconductor NM24CF04 4096-Bit (512 x 8) CMOS Serial Nonvolatile Memory. These memory chips provide the possibility of implementing non-volatile computer main memory. Recent studies [Ref. 1] have demonstrated the feasibility of directly interfacing FECM to a microprocessor memory bus.

### 1. FECM Theory of Operation

FECM are constructed using a modified silicon CMOS semiconductor process. Figure 1 shows a typical structure for a simple ferroelectric capacitor [Ref. 2]. A commonly used ferroelectric material is lead zirconate titanate (PZT). The PZT dielectric is deposited between two electrode layers, forming a capacitor. The capacitor is charged when written to and the charge is retained for greater than ten years in the case of the NM24CF04 FECM. The cell is read by applying a voltage across the capacitor and observing the current pulse generated.

A small current pulse indicates a '1' is stored in the cell while a large current pulse indicates a '0' is stored in the cell. It should be noted that the read operation is a "destructive read" and the data must be written back to the device.

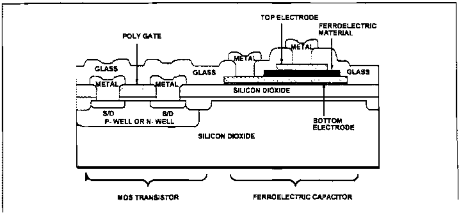

**Figure 1** Ferroelectric Capacitor Device, [Ref.2].

Current technology has demonstrated several advantages and favorable characteristics of FECM [Ref. 3], including: 1) non-volatility, 2) long retention (>10 years @ 25°C), 3) endurance ($10^{10}$ write cycles), and 4) radiation hardness (>$10^{6}$ RADs(Si/sec)). Therefore, FECM's are ideal for many military and space based systems.

**2. National Semiconductor's NM24CF04**

The NM24CF04 is fabricated with advanced CMOS ferroelectric technology [Ref. 4]. The device provides a 4096-bit nonvolatile memory, internally organized as two 256 x 8-bit pages. It uses a serial interface on a two-wire bus with practically unlimited erase/write cycles and supports a bidirectional bus oriented protocol. When power is removed, the data remains stored in the ferroelectric cells. Figure 2 shows a connection diagram for the device.

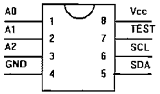

**Figure 2** NM24CF04 Pin Diagram, [Ref. 4].

Serial Data (SDA): SDA is a bidirectional pin used to transfer data into and out of the device. It is an open-drain output and may be wire ORed with any number of open-drain or open-collector outputs. A pull-up resistor is required.

Serial Clock (SCL): The SCL input is used to clock all data into and out of the device. Data states on the SDA line can change only during SCL LOW. SDA state changes during SCL HIGH are reserved for indicating start and stop conditions.

Address (A0): A0 is unused by the NM24CF04, however, it must be tied to GND to insure proper device operation.

Address (A1,A2): The Address inputs are used to set the least significant two bits of the six bit slave address. These inputs can be driven with logic or tied high or low. The four most significant bits of the slave address are tied to $V_{cc}$ or GND to form the "Device Type Identifier" nibble '1010'.

Pins A1 and A2 connected to +5 volts or ground to establish a hardwired address for the device. Two bits allow four different devices to be addressed.

### 3. Microprocessor System

The most commonly used 16-bit microprocessors today are the Intel 8086 series and the Motorola 68000. Previous studies [Ref. 1] used the Intel 8086 in the minimum mode. The thrust of this thesis research is centered around the Intel 8086 [Ref. 5], although the monitored signals could easily be converted to allow the use of the Motorola 68000 chip. The Minimum Mode System configuration is shown in Figure 3.

In the minimum mode, the CPU emits the bus control signals for memory. In the maximum mode, an 8288 Bus Controller assumes the responsibility of controlling all devices on the system bus. The CPU incorporates two separate processing units. These are the Execution Unit (EU) and the Bus Interface Unit (BIU). The EU executes instructions and the BIU fetches instructions, reads operands, and writes results. The two units operate independent of one another and are able, under most circumstances, to extensively overlap instruction fetch with execution. The 8086's instruction stream queue can store up to six instruction bytes. When two or more bytes of the 6-byte instruction queue are empty and the EU does not require the BIU to perform a bus cycle, the BIU executes instruction fetch cycles to refill the queue. The BIU normally fetches two bytes unless the fetch is from the odd address,

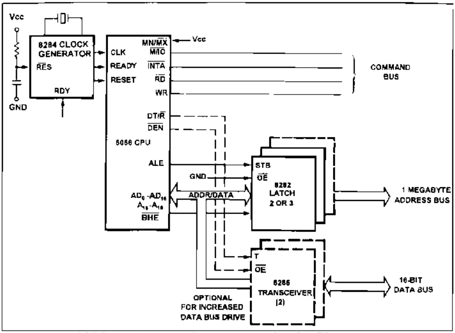

**Figure 3** 8086 Minimum Mode System, [Ref. 5].

where the BIU only fetches the byte from the odd address. The BIU is responsible for executing all external bus cycles to read data from memory or write data to memory. All bus cycles consist of a minimum of four clock cycles or "T-states" know as $T_1$, $T_2$, $T_3$, and $T_4$. Figure 3 shows a typical BIU bus cycle.

A bus cycle consists of a chain of events in which the address of memory location is output, then a read or write control signal is presented, followed by the data in a write operation. The addressed device accepts the data on a write cycle or places data on the bus during a read operation.

For slow devices, such as the NM24CF04, wait states must be inserted between $T_3$ and $T_4$. During a wait state, the data on the bus remains unchanged. Upon completion of the cycle, memory latches data written or the BIU removes data read.

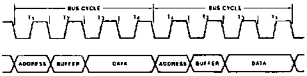

**Figure 4** Typical BIU Bus Cycle, [Ref. 1].

**a. *Write Bus Cycle***

Figure 4 illustrates the basic write bus cycle. The CPU places the write address on the multiplexed address/data bus during $T_1$. During state $T_1$, Address Latch Enable (ALE) is asserted high to latch the write address into a register. The register contents can be decoded by combinational logic to generate the necessary chip select for memory.

During state $T_2$, the CPU places the write data on the multiplexed bus (AD15 - AD0), and WR is asserted low to indicate to memory that a write operation is requested. The *READY* signal is used to force the CPU to insert wait states into the bus cycle. To insert a wait state, *READY* must be low prior to the end of state $T_3$. Figure 5 shows the timing relationship required to generate a wait state. Signal *READY*

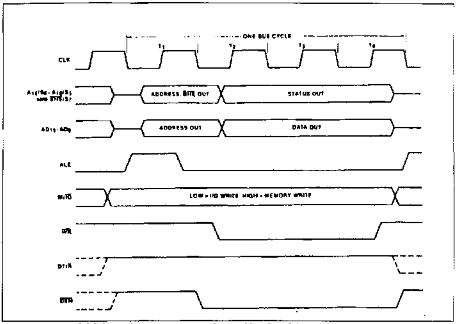

**Figure 5** 8086 Write Bus Cycle, [Ref. 1].

is produced by an 8284 Clock Generator and Driver shown in Figure 3. Signal *RDY* is produced by combinational logic in the interface device.

Signal Data Transmit/Receive (DT/R) controls the flow of data through banks of 8286 octal bus transceivers when operating in the minimum mode configuration. With the signal high, data can flow from the 8086 microprocessor onto the data bus and into the appropriate memory location. Data Enable

(*DEN*) is provided as an output enable for the data transceivers.

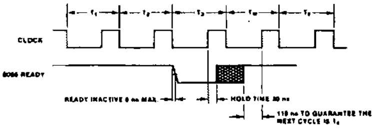

**Figure 6** Wait State Insertion, [Ref. 1].

### *b. Read Bus Cycle*

Figure 6 illustrates the 8086 Read Bus Cycle. The read cycle is very similar to the write cycle. The CPU places the read address on the multiplexed address/data bus during $T_1$. Again, during state $T_1$, *ALE* is asserted high to latch the write address into a register. The register contents is decoded to generate the necessary chip select for memory. Upon completion of $T_1$, the CPU floats the address/data bus in preparation for data to be received from memory. During state $T_2$, *RD* is asserted low to indicate to memory that a read operation is requested. Again, if reading from a slow memory device, the user is responsible for adding logic to control

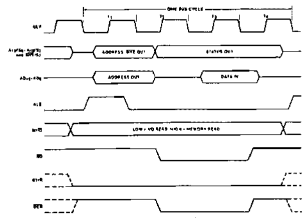

**Figure 7** 8086 Read Bus Cycle, [Ref. 1].

*READY* and generate wait states prior to state $T_3$. Signal *DT/R* is asserted low to configure transceivers in the receive mode. Signal *DEN* is asserted low to enable the transceivers to pass the received data to the CPU.

## B. OVERVIEW OF SYSTEM DESIGN

Figure 8 shows an overview of the system bus structure implemented in previous thesis research [Ref. 1].

### 1. Address and Data Bus Structure

Although the 8086 has the ability to address one megabyte of memory, the system described here employs 4K bytes of ferroelectric RAM (FERRO), 8K of ROM, and 8K of static RAM. The section of memory designated as FERRO is split into two

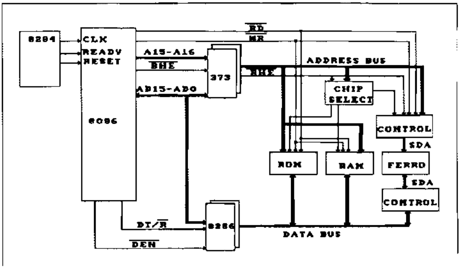

**Figure 8** System Bus Structure, [Ref. 1].

banks of 2K-bytes each. One bank connects the lower half of the data bus (D7-D0) and corresponds to even addresses (A0 = 0). The remaining bank connects the upper half of the data bus (D15-D8), and corresponds to odd addresses (A0 = 1). Address lines A1-A11 specify the particular byte in each bank. To perform a byte transfer to an even address, the 8086 specifies an address with A0 low and signals Bus High Enable (*BHE*) high. With *BHE* high, the upper bank of memory, or the odd byte of the word address, is disabled.

To perform a byte transfer to an odd address, the 8086 asserts *BHE* low and A0 high. This allows access to the upper bank while disabling any memory access of the lower bank or even byte. To perform a word transfer of 16 bits, the 8086 asserts both *BHE* and A0 low. This enables both memory banks and allows transfer of the entire data bus D15-D0 during one bus cycle.

## 2. Data Flow Overview

The objective of the design was to configure the NM24CF04 as a functional block of main memory without requiring any special actions on the part of the 8086 microprocessor. The NM24CF04 operates at a frequency of 100 KHz, thus requiring the wait states to be inserted whenever the ferroelectric memory was addressed.

The CONTROL logic of Figure 8 monitors five sets of signals: *RD, WR, BHE,* A0-A11, and D0-D15. When the 8086 requests a read operation from ferroelectric memory, it places the read address on the address bus and indicates a read request by asserting RD low. The CONTROL logic block takes over by initiating microprocessor wait states and performing the required tasks to retrieve information from the appropriate bank of ferroelectric memory. Control tasks performed include identifying and selecting the correct chip, converting the parallel address to a serial address, transmitting the address to memory serially, receiving the data from memory serially and transferring the received information into parallel form, latching the data, and releasing control to the microprocessor. Writing to ferroelectric memory is very similar.

When the microprocessor writes data to ferroelectric memory, the write address is placed on the address bus, data on the data bus, and a write operation is indicated by asserting WR low. Again, the CONTROL block intercedes to perform data translations and input/output functions necessary to store data into the appropriate address.

## C. RESEARCH GOALS

To extract a portion of a previously designed SSI/MSI interface [Ref. 1] and incorporate it in a single chip design using CMOS VLSI. Keeping the implementation as general as possible is a primary consideration. Figure 9 shows a block diagram of the extracted circuity. The Main Finite State

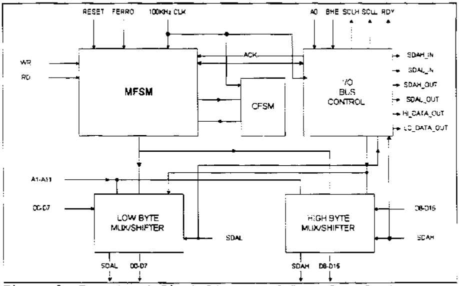

**Figure 9** Functional Block Diagram of Interface Device.

Machine (MFSM) monitors signals *WR*, *RD*, *FERRO*, and *ACK*. Signals *WR*, and *RD* were described in Section A. of this chapter. Signal *FERRO* is a chip select generated by external combinational logic. Signal *ACK* is generated by the I/O BUS CONTROL by monitoring the Serial Data Bus Low (SDAL) and Serial Data Bus High (SDAH) for the Acknowledge (*ACK*)

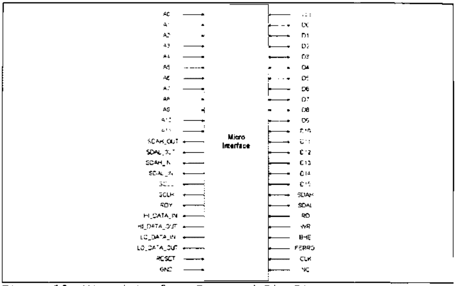

**Figure 10** Microinterface Proposed Pin Diagram.

generated by the NM24CF04. The Counter Finite State Machine (CFSM) maintains a count to control the number of shift operations performed by the shift registers contained in the LOW and HIGH BYTE MUX/SHIFTERs. The I/O BUS CONTROL generates clock and control for both internal and external bus control. LOW and HIGH BYTE MUX/SHIFTERs provide serial and parallel I/O to the address, parallel, and serial data busses. Figure 10 shows the proposed chip pin diagram. Signals not yet discussed are:

Address (*A0-A11*): Address bus lines from 74LS373 octal latches in Figure 8. *A10* and *A11* select the desired chip within a bank. *A9* selects the desired page within a chip. *A1-A8* selects the specific byte within the page.

Clock (*CLK*): A 100KHz clock is used to clock the "Microinterface" chip circuitry and generate *SCLL* and *SCLH* to clock the ferroelectric memory chips.

Serial Data Bus High/Low Out (*SDAH_OUT/SDAL_OUT*): These pins are used if an external SDA bus driver is used. They can be used to enable data onto a serial data bus with a 74LS241 Line Driver, as shown in Figure 57.

High/Low Data Bus Out (*HI/LO_DATA_OUT*): These pins are used if an external data bus driver is used. They can be used to enable data onto a parallel data bus with a tri-state Line Driver.

Serial Clock High/Low (*SCLH/SCLH*): The clock output that connects to the SCL pin of the respective NM24CF04 memory chip.

Ready (*RDY*): The output pin that connects to the RDY pin of the 8284 Clock Generator shown in Figure 3, to generate *READY*.

Serial Data Bus High/Low In *(SDAH_IN/SDAL_IN)*: Used to enable a tri-state Line Driver if the circuit is configured with external drivers as shown in Figure 57.

Data Lines *(D0-D15)*: These pins connect to the data bus from the Transceivers shown in Figure 8.

## D. REQUIRED HARDWARE AND SOFTWARE TOOLS

The design and layout of the Microinterface chip was done on the Naval Postgraduate School's Sun SPARCstations using the VLSI CAD tool package created by the University of California at Berkeley.

### 1. Sun SPARCstation

The Sun Microsystems SPARCstation II is a desktop workstation that offers high-speed color graphics. Operating at 40-MHz, the system is equipped with 32 MB of RAM, 424 MB of internal fixed disk storage, and mounts several large file systems from a remote server.

### 2. Magic

Magic is an interactive editor for creation of Very Large Scale Integration (VLSI) circuit layouts. This program runs on many UNIX based systems, for example, the Sun SPARCstation with an integrated color display. Using Magic, the designer can create basic cell layouts and combine them into larger structures, or even complete integrated circuit layouts. This program has a built in design rule checker that constantly checks layouts during creation to ensure that layout rules are obeyed for the particular technology being used. To allow a means of interfacing with other programs, Magic allows the user to extract the created circuits for use by other programs. Magic only permits Manhattan designs, which are designs with horizontal or vertical edges, no diagonal or curved structures are allowed. The further attributes of Magic are numerous, and reference to the user's manual is recommended for additional descriptions of its abilities [Ref. 6].

### 3. **Peg**

The PLA Equation Generator compiles finite state machines to generate a working PLA. It takes a high level description of a finite state machine and translates it into the logic equations required to implement a specific design. Peg generates logic equations that follow the Moore model of a finite state machine. This implies that the outputs generated by the finite state machine depend only on the current state of the machine. Inputs can be provided to cause transitions to specific states. The Peg command used was:

```
peg  infile > outfile
```

This allowed for the infiles, which in this case were mfsm.peg and cfsm.peg, to be processed into the equations required to design the heart of the extracted circuit. Included in

Appendix A. are copies of the peg input and output files [Ref. 6].

### 4. SPICE3C1

SPICE3C1 is an updated version of SPICE which is a general purpose circuit simulation program. It can perform dc analysis, as small-signal analysis, and transient analysis. SPICE can also be used to provide extensive plotting of circuit parameters and circuit behavior using its plot command. SPICE is a tremendously powerful tool in observing how circuits are acting, but it has a limitation in that large circuits take an extremely long time to process. SPICE was used mainly for power estimations and transient behavior [Ref. 7].

### 5. Esim

Esim is an event-driven switch-level simulator for NMOS and CMOS circuits. It can be entered after the circuit has been extracted from Magic. Esim is used to watch various nodes, to set or reset nodes, and to simulate the logical operation of the circuit. The watched nodes can then be inspected and the circuit evaluated. There are numerous commands and options that may be employed in the simulation and the reader is directed to the reference material for further clarification [Ref. 6].

## E. THESIS STRUCTURE

Chapter I introduces the need for ferroelectrics, the NM24CF04, the 8086 microprocessor, and details of previous research. System timing considerations are discussed. Chapter II contains a description of circuit extraction considerations that were necessary to develop the design of the interface between the NM24CF04 and the 8086 microprocessor. VLSI layout considerations are described. Chapter III describes basic design method used to for each major section of the chip. An estimation of static and dynamic power dissipation is made. Chapter IV discussion of testing objectives, procedure, and results. Chapter V summarizes the study and includes further recommendations.

# II. DESIGN CONSIDERATIONS

The general interface device design began with careful consideration of the system bus structure shown in Figure 8. The functional blocks shown in Figure 9 described the implementation of the single chip design. These blocks were chosen to incorporate as much circuitry as possible from the previous SSI/MSI interface implementation [Ref. 1], and still maintain versatility for future adaptation to other microprocessor applications. The Magic layout editor requires that a filename be used for cell creation. The name "Microinterface" is selected to describe this chip. Further references to the entire chip are made using this name.

## A. MAIN FINITE STATE MACHINE (MFSM)

The MFSM generates 15 different states that are decoded to provide control signals to the multiplexers and shift registers in the LOW and HIGH BYTE MUX/SHIFTER blocks. Outputs include states *a* - *o*, shift register function controls *s0* and *s1*, multiplexer byte selectors *muxa* and *muxb*, and counter control signals *counter_clr* and *counter_en*. The design was extracted directly from the SSI/MSI design [Ref. 1], and a detailed description of the operation may be found in that reference.

## B. COUNTER FINITE STATE MACHINE (CFSM)

The CFSM monitors the output states and counter control signals from the MFSM, and counts from zero to seven when enabled. While the CFSM is counting, the MFSM is dormant and is reactivated when the CFSM output signal *cntr7* is asserted.

## C. HIGH/LOW BYTE MUX/SHIFTER

The LOW and HIGH BYTE MUX/SHIFTERs are identical functional blocks. Each block consists of a 4-to-1 byte multiplexer and shift register. The most important function performed by this block is the parallel to serial conversion of information and vice versa. The high and low bits of the shift register connect to the output and input of the appropriate Serial Data Bus (SDA). The LOW BYTE MUX/SHIFTER connects to the Serial Data Bus Low (SDAL), and the HIGH BYTE MUX/SHIFTER connects to the Serial Data Bus High (SDAH).

Four possible bytes may be selected by the multiplexer. Figure 11 shows the byte formats. For simplicity, only the low-byte format is shown. The high-byte format only differs in byte 2 and would be D8-D15 vice D0-D7. Byte 0 selects a particular chip in a bank. Byte 1 addresses one of 512 memory locations on a specific chip. Byte 2 is the data to be written if the operation is a write. Byte 3 is only used during the read operation and is called a "dummy read". Bit R/W of Byte 0 indicates the desired operation to the NM24CF04. When set to one, a read operation is selected, when set to zero, a write

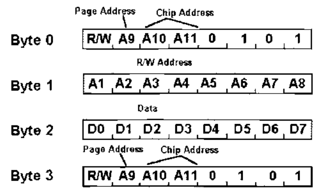

**Figure 11** Byte Formats.

operation is selected. The R/W position is hard wired to a one in Byte 3. As discussed in Chapter I, the high nibble '1010' forms the "Device Type Identifier".

## D. I/O BUS CONTROL

The I/O Bus Control Block is a catch all group of circuits that generates the remaining required signals. These signals include the required shift register clocks, SDA/DATA enables, NM24CF04 Start/Stop signals, and the Acknowledge enable used to capture *ACK*.

## E. INTERFACE DEVICE OVERALL CHIP LAYOUT

Figure 12 is a floor plan of the "Microinterface" IC showing the overall layout of the major cells. For simplicity, the required cell interconnects, and chip pads are omitted. Reducing the space required for cell interconnects is a major consideration when planning a chip layout. This particular layout structure allowed for minimum distance between cells for greater compactness and less wasted chip space. The resultant chip is nearly square measuring four millimeters on a side. Detailed Magic cell layouts are provided in Appendix B.

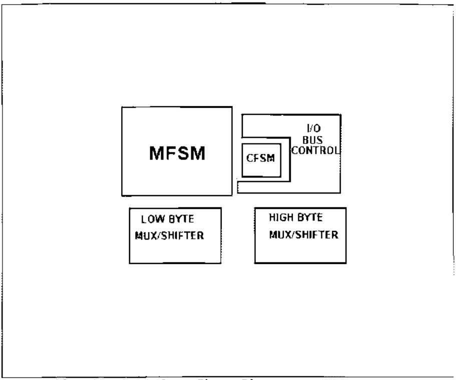

**Figure 12** Microinterface Floor Plan.

# III. IMPLEMENTATION

The basic logic design of each of the cells used in the "Microinterface" interface device are shown. This will provide insight into the design characteristics of each of the cells. The basic cell layouts are included in Appendix B so that the reader can see how the cell descriptions given appear in a Magic layout.

## A. CELL CONVERSION

The cell plots in the appendix were created using the Magic function cif, which creates a file cell.cif [Ref. 6]. The .cif file is then converted by the routine cif2ps. In order to include these files in the thesis, further conversion to a .pcx file was needed. The following conversion is available on alioth at the computer center as follows:

```
pstoppm filename.ps | imconv -ppm - -pcx filename.pcx
```

Note: This is a general conversion utility. To convert to some other standard format, replace pcx with the extension of the type of graphics you want to convert to, i.e. tiff, gif, eps, etc.

## B. MAIN/COUNTER FINITE STATE MACHINES (MFSM/CFSM)

The MFSM and CFSM were developed from the original Peg programs used in the SSI/MSI design [Ref. 1]. The programs may be seen in Appendix A. Peg generates equations in a minterm fashion. The equations are expressed as the sum of products and are easily converted using DeMorgan's theorem. The resultant equations can be directly implemented using two stage NAND-NAND logic. As an example of the method used, the combinational logic development for State0 is detailed starting with the Peg output equation:

Step 1: Peg equation

```
  OutSt0*=
     (!RESET& InSt0*& InSt1*&!InSt2*)|
     (!RESET& InSt0*&!InSt1*& InSt2*)
     (!RESET& InSt0*&!InSt1*&!InSt2*& InSt3*)
     (!RESET&!InSt0*& InSt1*& InSt2*& InSt3*)|
     (!RESET& cntr7& wr&!InSt0*& InSt1*&!InSt2*&!InSt3*)
```

The symbols '|', '!', and '&' represent OR, NOT, and AND operations, respectively. Converting the equation to a more manageable format:

$$\begin{aligned} State0 = {} & (\overline{RESET}\cdot Q0\cdot Q1\cdot \overline{Q2}) + (\overline{RESET}\cdot Q0\cdot \overline{Q1}\cdot Q2) + \\ & (\overline{RESET}\cdot Q0\cdot \overline{Q1}\cdot \overline{Q2}\cdot Q3) + (\overline{RESET}\cdot \overline{Q1}\cdot Q2\cdot Q3) + \\ & (\overline{RESET}\cdot cntr7\cdot wr\cdot \overline{Q0}\cdot Q1\cdot \overline{Q2}\cdot \overline{Q3}) \end{aligned}$$

Step 2: Apply DeMorgan's theorem

$$\begin{aligned} State0 = {} & \overline{\overline{(\overline{RESET}\cdot Q0\cdot Q1\cdot \overline{Q2})}\cdot \overline{(\overline{RESET}\cdot Q0\cdot \overline{Q1}\cdot Q2)}\cdot} \\ & \overline{\overline{(\overline{RESET}\cdot Q0\cdot \overline{Q1}\cdot \overline{Q2}\cdot Q3)}\cdot \overline{(\overline{RESET}\cdot \overline{Q1}\cdot Q2\cdot Q3)}\cdot} \\ & \overline{\overline{(\overline{RESET}\cdot cntr7\cdot wr\cdot \overline{Q0}\cdot Q1\cdot \overline{Q2}\cdot \overline{Q3})}} \end{aligned}$$

The equation is now in the NAND-NAND form and can be directly implemented with one seven input, two five input, and three four input NAND gates.

Step 3: Logical schematic

The logical schematic for State0 is shown in Figure 13.

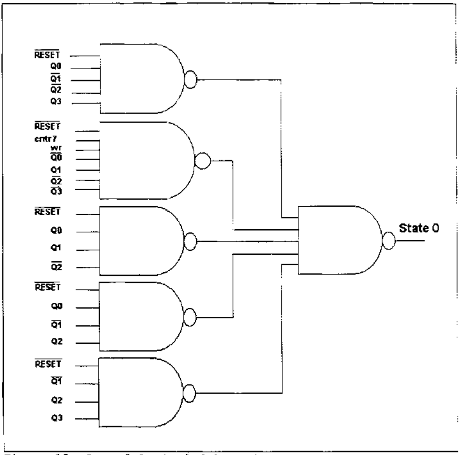

**Figure 13** State0 Logical Schematic.

The logical schematic can now be used as a guide for constructing the Magic layout.

Step 4: Magic Layout

A collection of elementary Magic cells were designed for use in a hierarchial design of the larger cells. The Magic layout of State0 with inverters added is shown in Figure 14. Detailed Magic layouts can be seen in Appendix B.

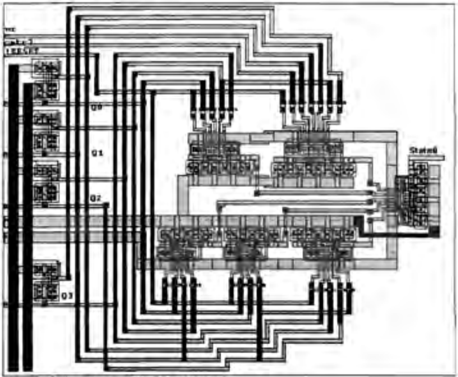

**Figure 14** State0 Magic Layout.

## C. HIGH/LOW BYTE MUX/SHIFTER

The HIGH/LOW BYTE MUX/SHIFTERs consist of two cells each. A 4-to-1 byte multiplexer (MUX) and an 8-bit shift register. The 4-to-1 byte MUX consists of a bank of eight 4-to-1 bit multiplexers. Bytes are selected by *muxa* and *muxb* as discussed in Chapter 2. The shift register is a bidirectional, parallel-in, serial-out, serial-in, parallel-out, general purpose shift register. Both the MUX and shift register can be found in all basic digital design books.

## D. I/O BUS CONTROL

The I/O BUS CONTROL was extracted from the EPLD programs used in the SSI/MSI implementation [Ref. 1]. The design method paralleled that used in the MFSM design. Extracted equations accompany the respective cell layouts found in Appendix B.

## E. POWER ESTIMATES

Electromigration is the transport of metal ions through a conductor that results from excessive current density in the conductor. In order to determine the minimum size of power and ground conductors, an estimate of current density must be made. If the maximum current density, for a specific size conductor, is exceeded, the conductor may weaken and eventually blow like a fuse [Ref. 8].

SPICE3C1 was used for dc and ac analyses to determine power consumption of each circuit.

## 1. Static Power Dissipation

The static power consumption is defined as worst-case leakage current multiplied by Vdd in any specific device. The results for each of the elementary cells used in the Microinterface device are shown in Table I.

**Table I** STATIC POWER DISSIPATION.

| CELL | CELL POWER | No. of CELLS | TOTAL |
|---|---|---|---|
| BUFF | 5.50 μW | 25 | 138 μW |
| INV1 | 146 nW | 122 | 17.8 μW |
| INV4 | 5.60 μW | 83 | 465 μW |
| NAND2 | 495 nW | 143 | 70.8 μW |
| NAND3 | 854 nW | 91 | 77.7 μW |
| NAND4 | 861 nW | 38 | 32.7 μW |
| NAND5 | 890 nW | 14 | 12.5 μW |
| NAND6 | 253 nW | 16 | 4.05 μW |
| NAND7 | 404 nW | 3 | 1.21 μW |
| NAND8 | 202 nW | 1 | 202 nW |
| NAND10 | 1.04 μW | 1 | 1.04 μW |
| NOR2 | 356 pW | 4 | 1.42 nW |
| NOR3 | 369 pW | 2 | 738 pW |
| NOR4 | 0.585 pW | 2 | 1.17 pW |
| NOR5 | 0.585 pW | 2 | 1.17 pW |
| XOR2 | 55.5 nW | 8 | 444 nW |
| TOTALS | 16.3 μW | 555 | 821 μW |

## 2. Dynamic Power Dissipation

The dynamic power dissipation is defined as the current drawn by the cell during transition multiplied by Vdd. The total power dissipation for the device is then calculated by multiplying by the number of devices used of each type. The results may be seen in Table II.

**Table II** DYNAMIC POWER DISSIPATION.

| CELL | CELL POWER | No. of CELLS | TOTAL |
|---|---|---|---|
| BUFF | 9.25 mW | 25 | 231 mW |
| INV1 | 2.57 mW | 122 | 314 mW |
| INV4 | 3.81 mW | 83 | 316 mW |
| NAND2 | 2.45 mW | 143 | 350 mW |
| NAND3 | 2.46 mW | 91 | 224 mW |
| NAND4 | 2.48 mW | 38 | 94.2 mW |
| NAND5 | 2.54 mW | 14 | 35.6 mW |
| NAND6 | 2.52 mW | 16 | 40.3 mW |
| NAND7 | 2.35 mW | 3 | 7.05 mW |
| NAND8 | 2.40 mW | 1 | 2.40 mW |
| NAND10 | 2.40 mW | 1 | 2.40 mW |
| NOR2 | 1.10 mw | 4 | 4.40 mW |
| NOR3 | 1.18 mW | 2 | 2.36 mW |
| NOR4 | 1.18 mW | 2 | 2.36 mW |
| NOR5 | 735 µW | 2 | 1.47 mW |
| XOR2 | 994 µW | 8 | 7.95 mW |
| TOTALS | 40.mW | 555 | 1.64 W |

## F. MICROINTERFACE PHYSICAL IMPLEMENTATION

After the actual chip is constructed, physical wire-up of a test circuit will be required. Possible circuits can be seen in Appendix D. Two implementations are shown, and both show only the memory side of the circuit, the reader should refer to the microprocessor user's manual [Ref. 5] for circuit configuration in the minimum mode. First, Figure 56 shows the interface device connected in the most elementary manner. The chip is intended to connect directly to the SDA, Data, and Address busses with no external line drivers. If desired, the user may incorporate external line drivers and connect the circuit as shown in Figure 57. The device provides enable lines for each bus. In each case a chip select circuit must be provided by the user to select the main memory address where the ferroelectric memory is installed.

# IV. TESTING

All magic cells were tested with the Esim simulator discussed in Chapter I. Each major block of the hierarchy was, in turn, tested with Esim. The elementary cells were tested with SPICE to observe transient behavior. The operation of the chip may be fully verified by testing the MFSM, CFSM, High/Low Byte Mux/Shifter, larger cells of the I/O Bus Control, and a pad-to-pad check of the entire chip. Esim outputs of each of these may be seen in Appendix C.

## A. MAIN FINITE STATE MACHINE

The MFSM inputs include a clock (*CLK*), flip-flop preset (*PR*), flip-flop clear (*CLR*), read high/write low (*wr*), count to seven (*cntr7*), SDA acknowledge (*ack*), and chip select (*!ferro*). The outputs are states (*a-o*) and multiplexer inputs (*muxa, muxb, s0,* and *s1*). The circuit was stepped through all states and all outputs verified correct. After verifying proper operation of *PR* and *CLR*, they were tied to Vdd as they were used only for testing.

## B. COUNTER FINITE STATE MACHINE

The CFSM inputs are *counter_clr* and *counter_en*. The signal *counter_clr* occurs in states *b, d, f, k,* and *m* to clear the counter. The signal *counter_en* occurs in states *c, e, g, l,* and *n* to enable the counter. The state outputs of CFSM are available for use with the bus timing circuitry. In particular, state 1 is used to clear the *ack* capture circuit in the I/O Bus Control. The signal *cntr7* is output when the CFSM cycle is complete.

## C. HIGH/LOW BYTE MUX/SHIFTER

The High/Low Byte Mux/Shifter monitors *muxa*, *muxb*, *s0*, and *s1* control signals to perform the loading and shifting operations of the four bytes discussed in Chapter II. Correct operation with a representative bus cycle is shown in Appendix C.

## D. I/O BUS CONTROL

The I/O Bus Control section covers all remaining signals used to manipulate the Serial Clocks, Start/Stop condition, *ack* capture, *RDY*, and the enable lines for the external line drivers. Complete computer simulation using Esim may be seen in Appendix C.

## E. PAD-TO-PAD VERIFICATION

The pad-to-pad verification is the result of several iterations of testing of each major subcell. During this phase of testing, many problems were discovered. Some problems were easily rectified. However, the acknowledge enable (ACK_EN) would not function using a bidirectional SDAL pin configuration. Redesign of the ACK_EN capture circuit was required. Proper operation was finally achieved. Another significant problem encountered was the numbering order of the address/data bus into the shift register. The NM24CF04 manufacturer's documentation [Ref. 4] is not clear as to which bit should be shifted into the device first. The final design used the bit order determined in the previous MSI/SSI study. The most significant bit of the respective address or data byte is shifted into the NM24CF04 first.

Referring to Appendix C, Section M, a typical read cycle can be seen. Ferroelectric memory address $FF_{16}$ is addressed in both the low and high banks. The MFSM states are monitored for clarity of operation. Control bytes 0 and 3 can be seen on the SDAL/H bus during states c and l. The address $FF_{16}$ is on the bus during state e. The NM24CF04 responses with the data bytes $01010101_2$ for D0-D7 and $10101010_2$ for D8-D15 during state o. The typical write cycle is verified in a similar manner.

The pad-to-pad verification required careful consideration of the operation of the bidirectional pins. It should be noted that the SDAL/H bus pins were only driven when the NM24CF04 would be expected to respond. Otherwise, a 'don't care' condition was applied to allow monitoring of the Microinterface output.

# V. CONCLUSIONS AND RECOMMENDATIONS

## A. CONCLUSIONS

The "Microinterface" interface circuit has been designed for implementation on a single chip. Although "Microinterface" was designed for a standard CMOS process, the single chip implementation without EPLD's and the shear reduction of size is a new contribution to microprocessor memory system technology. The original SSI/MSI implementation required 59 chips while implementation using the "Microinterface" chip would require only 36 chips.

This research project provides the necessary follow-on study for VLSI implementation of the required digital circuitry to utilize ferroelectric memory as a portion of main microprocessor memory. The use of ferroelectric technology would provide a radiation hardened and nonvolatile, yet modifiable, area of memory where mission parameters or new programs could be stored. This VLSI implementation would more easily allow the incorporation of the NM24CF04 memory device in a microprocessor based system. The use of ferroelectric memory in any system would combine the strengths of current technologies: the flexibility of RAM and the nonvolatility of ROM.

The ferroelectric memory device is inherently slow due to the serial-access design. Conversion of the address/data from parallel to serial format requires the microprocessor to wait an inordinate amount of time for memory access. As the ferroelectric technology evolves, what is really needed is a ferroelectric integrated circuit that is accessed by means compatible with current RAM accessing techniques. The availability of a parallel accessed ferroelectric memory would vastly improve read and write cycle times.

## B. RECOMMENDATIONS

1) Fabricate chip and test in a circuit similar to the original SSI/MSI circuit.

2) Design and implement circuitry required for use with the Motorola 68000 microprocessor.

## APPENDIX A:  PEG PROGRAMS

### A. MAIN FINITE STATE MACHINE INPUT

```
INPUTS:      RESET ferro cntr7 ack wr;
OUTPUTS:     s1 s0 muxa muxb a b c d e f g h i j k l m n o;

start:       ASSERT a;
             IF NOT ferro THEN stateb ELSE LOOP;

stateb:      ASSERT b s1 s0;
             GOTO statec;

statec:      ASSERT c s0;
             IF cntr7 THEN stated ELSE LOOP;

stated:      ASSERT d muxa s0 s1;
             IF ack THEN statee ELSE LOOP;

statee:      ASSERT e s0;
             CASE (cntr7 wr)
             1 0 => statef;
             1 1 => statej;
             ENDCASE => statee;

statef:      ASSERT f s0 s1 muxb;
             IF ack THEN stateg ELSE LOOP;

stateg:      ASSERT g s0;
             IF cntr7 THEN stateh ELSE LOOP;

stateh:      ASSERT h;
             GOTO statei;

statei:      ASSERT i;
             GOTO start;

statej:      ASSERT j s0 s1 muxa muxb;
             IF ack THEN statek ELSE LOOP;

statek:      ASSERT k s0 s1 muxa muxb;
             GOTO statel;
```

```
statel:       ASSERT l s0;
              IF cntr7 THEN statem ELSE LOOP;

statem:       ASSERT m s0;
              IF ack THEN staten ELSE LOOP;

staten:       ASSERT n s0;
              IF cntr7 THEN stateo ELSE LOOP;

stateo:       ASSERT o;
              GOTO start;
```

**B. MAIN FINITE STATE MACHINE OUTPUT**

```
INORDER=
   RESET
   ferro
   cntr7
   ack
   wr
   InSt0*
   InSt1*
   InSt2*
   InSt3*;
OUTORDER=
   OutSt3*
   OutSt2*
   OutSt1*
   OutSt0*
   s1
   s0
   muxa
   muxb
   a
   b
   c
   d
   e
   f
   g
   h
   i
   j
   k
   l
   m
   n
   o;
OutSt3*=
     (!RESET&!cntr7& InSt0*& InSt1*&!InSt2*& InSt3*)|
      (!RESET& ack& InSt0*& InSt1*&!InSt2*&!InSt3*)|
      (!RESET&!cntr7& InSt0*&!InSt1*& InSt2*& InSt3*)
      (!RESET& InSt0*&!InSt1*& InSt2*&!InSt3*)|
      (!RESET&!ack& InSt0*&!InSt1*&!InSt2*& InSt3*)
      (!RESET& cntr7&!InSt0*& InSt2*&!InSt3*)|
      (!RESET&!ack&!InSt0*& InSt1*&!InSt2*& InSt3*)|
      (!RESET& cntr7&!InSt0*& InSt1*&!InSt2*&!InSt3*)|
      (!RESET&!ack&!InSt0*&!InSt1*& InSt2*& InSt3*)|
      (!RESET&!ferro&!InSt0*&!InSt1*&!InSt2*&!InSt3*);
OutSt2*=
      (!RESET& cntr7& InSt0*& InSt1*&!InSt2*& InSt3*)|
      (!RESET&!cntr7& InSt0*&!InSt1*& InSt2*& InSt3*)|
      (!RESET&!InSt1*& InSt2*&!InSt3*).
      (!RESET& ack& InSt0*&!InSt1*&!InSt2*& InSt3*)
      (!RESET&!InSt0*& InSt1*& InSt2*&!InSt3*)|
      (!RESET& ack&!InSt0*& InSt1*&!InSt2*& InSt3*)|
      (!RESET&!ack&!InSt0*&!InSt1*& InSt2*& InSt3*)|
      (!RESET&!InSt0*&!InSt1*&!InSt2*& InSt3*);
OutSt1*=
      (!RESET& InSt0*& InSt1*&!InSt2*)|
      (!RESET& cntr7& InSt0*&!InSt1*& InSt2*& InSt3*)
      (!RESET&!InSt0*& InSt1*& InSt2*&!InSt3*)
      (!RESET&!InSt0*& InSt1*&!InSt2*& InSt3*)
      (!RESET& cntr7&!wr&!InSt0*& InSt1*&!InSt2*&!InSt3*)|
      (!RESET&!cntr7&!InSt0*& InSt1*&!InSt2*&!InSt3*)|
      (!RESET& ack&!InSt0*&!InSt1*& InSt2*& InSt3*);
OutSt0*=
      (!RESET& InSt0*& InSt1*&!InSt2*)|
      (!RESET& InSt0*&!InSt1*& InSt2*)|
      (!RESET& InSt0*&!InSt1*&!InSt2*& InSt3*)|
      (!RESET&!InSt0*& InSt1*& InSt2*& InSt3*)|
      (!RESET& cntr7& wr&!InSt0*& InSt1*&!InSt2*&!InSt3*);
s1=
      ( InSt0*&!InSt1*& InSt2*&!InSt3*)|
      (!InSt1*&!InSt2*& InSt3*)|
      (!InSt0*& InSt1*&!InSt2*& InSt3*)|
      (!InSt0*&!InSt1*& InSt2*& InSt3*);
s0=
      ( InSt0*& InSt1*&!InSt2*)|
      (!InSt1*& InSt2*)|
      (!InSt1*&!InSt2*& InSt3*)|
      (!InSt0*& InSt1*&!InSt3*)|
      (!InSt0*& InSt1*&!InSt2*& InSt3*);
muxa=
      ( InSt0*&!InSt1*& InSt2*&!InSt3*)|
      ( InSt0*&!InSt1*&!InSt2*& InSt3*)|
      (!InSt0*&!InSt1*& InSt2*& InSt3*);
muxb=
      ( InSt0*&!InSt1*& InSt2*&!InSt3*)|
      ( InSt0*&!InSt1*&!InSt2*& InSt3*)|
      (!InSt0*& InSt1*&!InSt2*& InSt3*);
```

```
a=
     (!InSt0*&!InSt1*&!InSt2*&!InSt3*);
b=
     (!InSt0*&!InSt1*&!InSt2*& InSt3*);
c=
     (!InSt0*&!InSt1*& InSt2*&!InSt3*);
d=
     (!InSt0*&!InSt1*& InSt2*& InSt3*);
e=
     (!InSt0*& InSt1*&!InSt2*&!InSt3*);
f=
     (!InSt0*& InSt1*&!InSt2*& InSt3*);
g=
     (!InSt0*& InSt1*& InSt2*&!InSt3*);
h=
     (!InSt0*& InSt1*& InSt2*& InSt3*);
i=
     ( InSt0*&!InSt1*&!InSt2*&!InSt3*);
j=
     ( InSt0*&!InSt1*&!InSt2*& InSt3*);
k=
     ( InSt0*&!InSt1*& InSt2*&!InSt3*);
l=
     ( InSt0*&!InSt1*& InSt2*& InSt3*);
m=
     ( InSt0*& InSt1*&!InSt2*&!InSt3*);
n=
     ( InSt0*& InSt1*&!InSt2*& InSt3*);
o=
     ( InSt0*& InSt1*& InSt2*&!InSt3*);
```

## C. COUNTER FINITE STATE MACHINE INPUT

```
INPUTS:      RESET;
OUTPUTS:     cntr7;

start:
     :
     :
     :
     :
     :
     :
     :       ASSERT cntr7;
             GOTO start;
```

## D. COUNTER FINITE STATE MACHINE OUTPUT

```
INORDER=
   RESET
   InSt0*
   InSt1*
   InSt2*;
OUTORDER=
   OutSt2*
   OutSt1*
   OutSt0*
   cntr7;
OutSt2*=
      (!RESET&!InSt2*);
OutSt1*=
      (!RESET& InSt1*&!InSt2*)|
      (!RESET&!InSt1*& InSt2*);
OutSt0*=
      (!RESET& InSt0*&!InSt2*)|
      (!RESET& InSt0*&!InSt1*& InSt2*)|
      (!RESET&!InSt0*& InSt1*& InSt2*);
cntr7=
      ( InSt0*& InSt1*& InSt2*);
```

APPENDIX B:  MAGIC CELL LAYOUTS

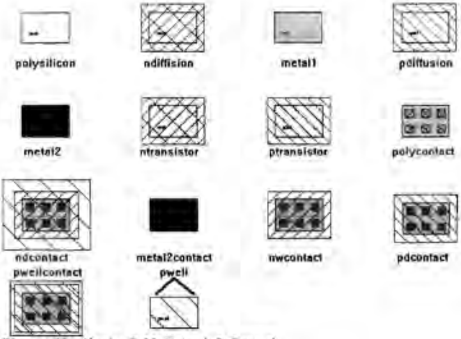

Figure 15  Magic Cell Material Legend.

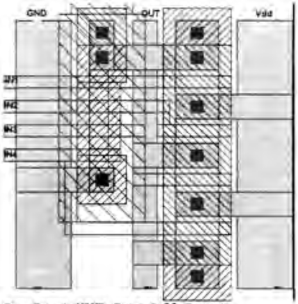

Figure 16  Two Input NAND Gate Cell Layout

OUT=IN1*IN2*IN3

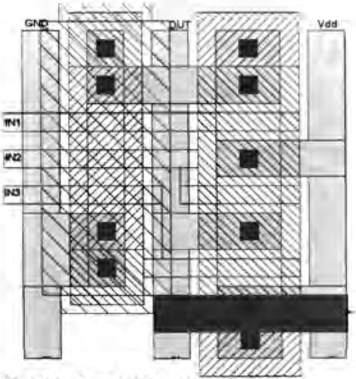

Figure 17  Three Input NAND Gate Cell Layout.

$VO = \overline{IN1 \cdot IN2 \cdot IN3 \cdot IN4}$

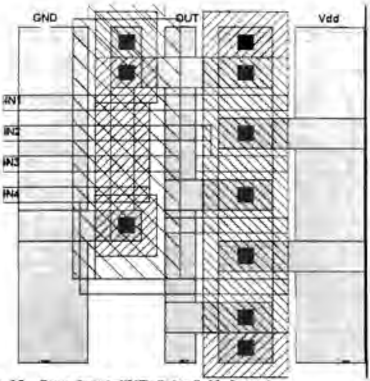

Figure 18 Four Input NAND Gate Cell Layout.

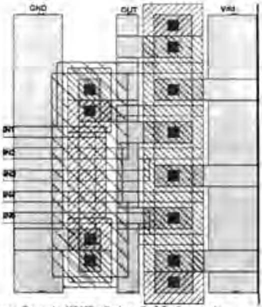

Figure 19  Five Input NAND Gate Cell Layout.

$$OUT = \overline{IN1 \cdot IN2 \cdot IN3 \cdot IN4 \cdot IN5 \cdot IN6}$$

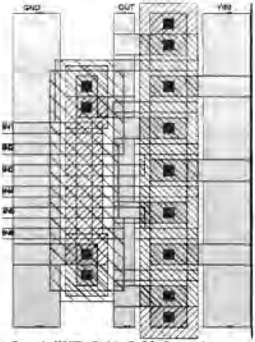

**Figure 20 Six Input NAND Gate Cell Layout.**

$$OUT = \overline{IN1 \cdot IN2 \cdot IN3 \cdot IN4 \cdot IN5 \cdot IN6 \cdot IN7}$$

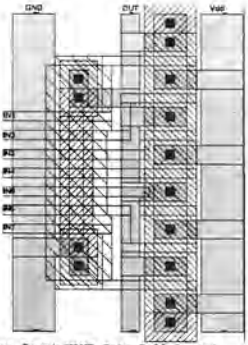

Figure 21  Seven Input NAND Gate Cell Layout.

$OUT = \overline{IN1 \cdot IN2 \cdot IN3 \cdot IN4 \cdot IN5 \cdot IN6 \cdot IN7 \cdot IN8}$

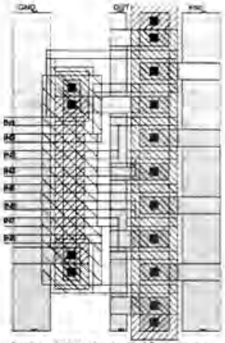

Figure 22  Eight Input NAND Gate Cell Layout.

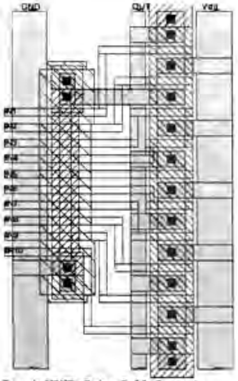

Figure 23 Ten Input NAND Gate Cell Layout.

$OUT=\overline{IN1+IN2}$

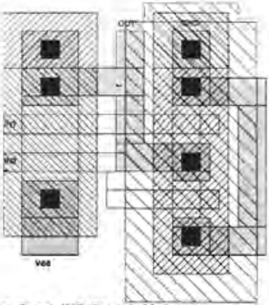

Figure 24  Two Input NOR Gate Cell Layout.

$$OUT=\overline{IN1+IN2+IN3}$$

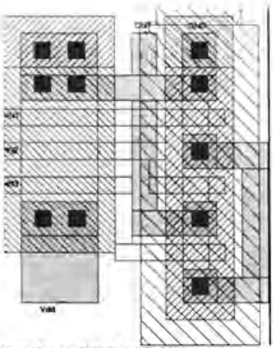

Figure 25  Three Input NOR Gate Cell Layout.

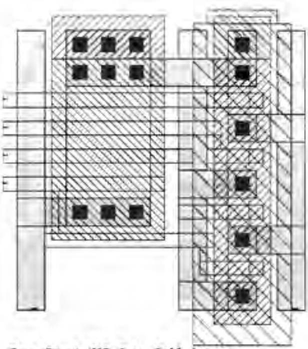
**Figure 26  Four Input NOR Gate Cell Layout.**

$OUT = \overline{IN1+IN2+IN3+IN4+IN5}$

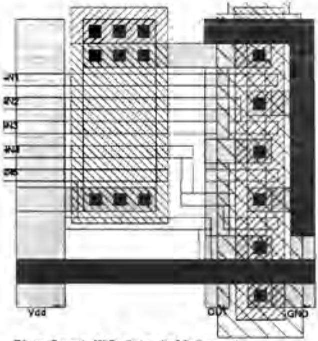

Figure 27  Five Input NOR Gate Cell Layout.

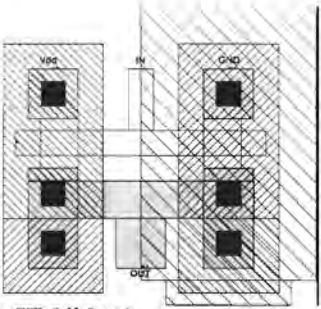

Figure 28  INV1 Cell Layout

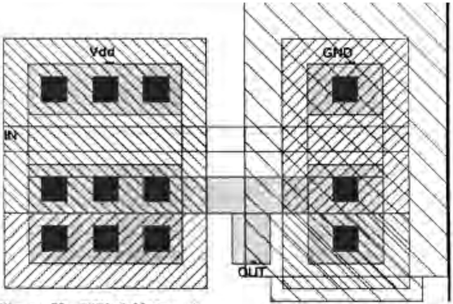

Figure 29  INV4 Cell Layout

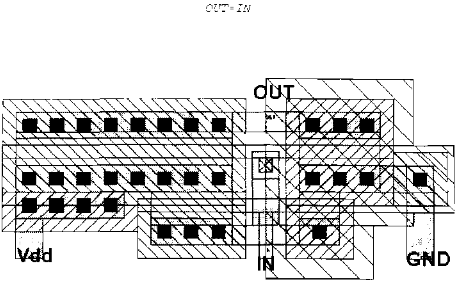

Figure 30  Buff Cell Layout.

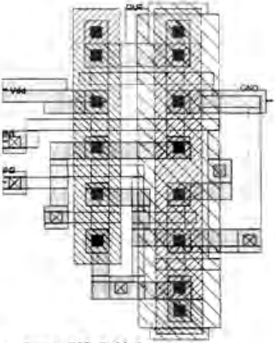

**Figure 31** Two Input XOR Cell Layout.

see Appendix A: for implementation equation of circuit abcd.

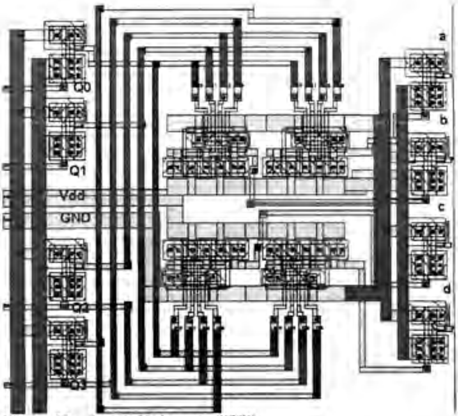

**Figure 32** abcd Cell Layout (MFSM).

See Appendix A. for implementation equation for circuit efgh.

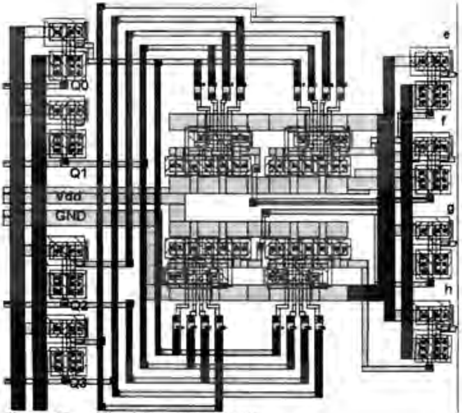

**Figure 33** efgh Cell Layout (MFSM).

See Appendix A. for implemented equation of circuit ijkl

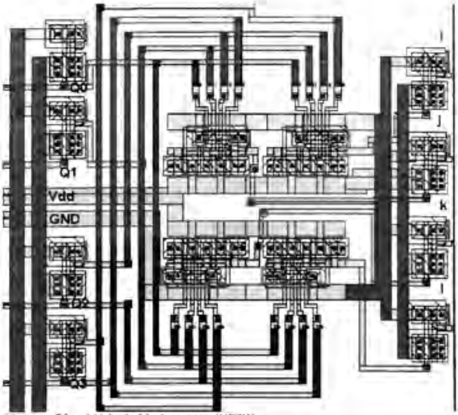

**Figure 34** ijkl Cell Layout (MFSM).

see Appendix A. for implementation equation for output mno.

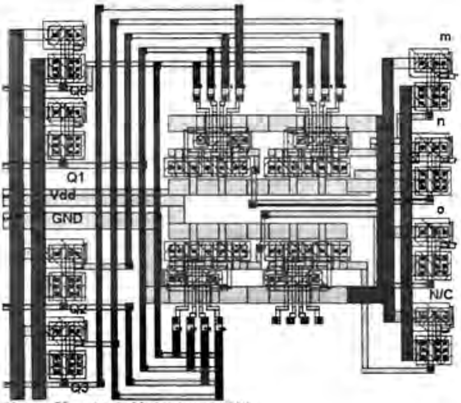

**Figure 35** mno Cell Layout (MFSM).

See Appendix A. for implementation equation for circuit State0.

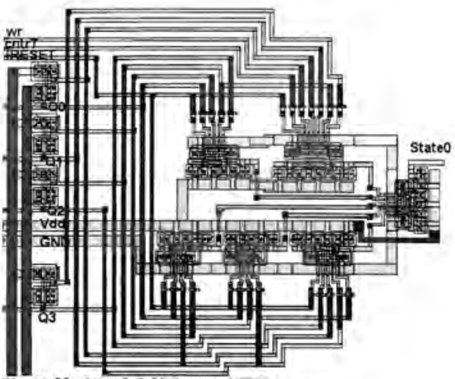

**Figure 36** State0 Cell Layout (MFSM).

See Appendix A. for implementation equation for circuit State1.

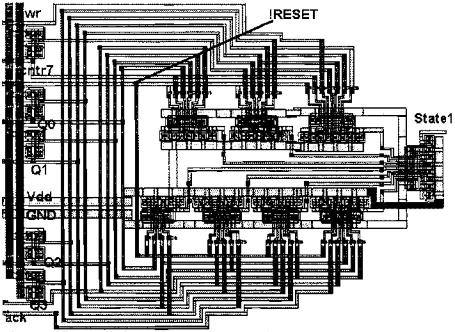

**Figure 37** State1 Cell Layout (MFSM).

See Appendix A. for implementation equation for circuit State2.

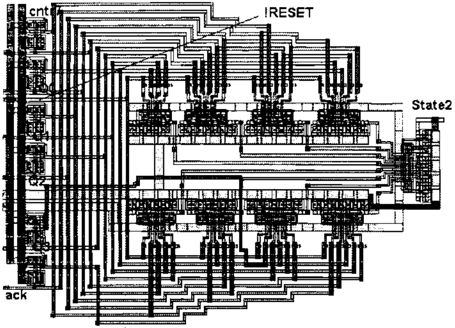

**Figure 38** State2 Cell Layout (MFSM).

See Appendix A. for implementation equation for circuit State3.

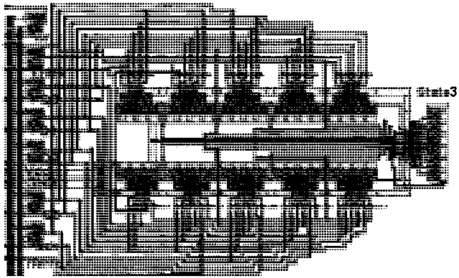

**Figure 39** State3 Cell Layout (MFSM).

See Appendix A, for implementation equation for circuit s1

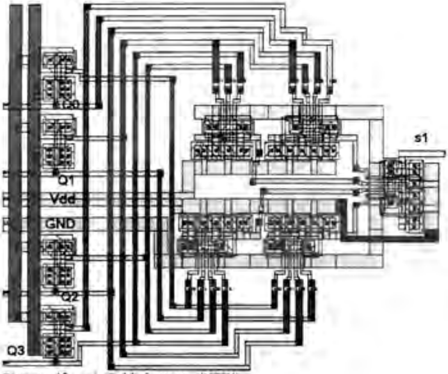

**Figure 40** s1 Cell Layout (MFSM).

See Appendix A. for implementation equation for circuit s0.

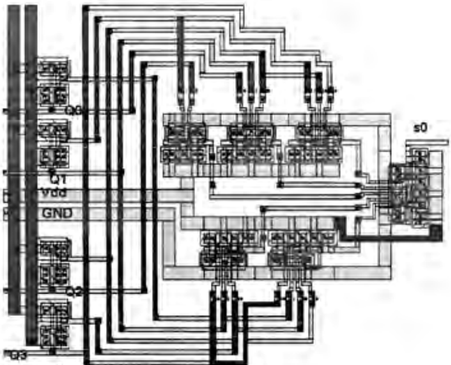

**Figure 41** s0 Cell Layout (CFSM).

See Appendix A. for implementation equation for circuit muxa_fsm (muxa).

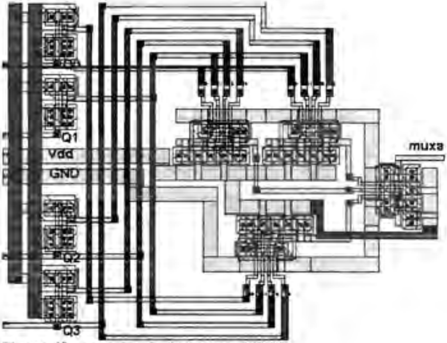

**Figure 42** muxa_fsm Cell Layout (MFSM).

See Appendix A. for implementation equation for circuit muxb_fsm (muxb).

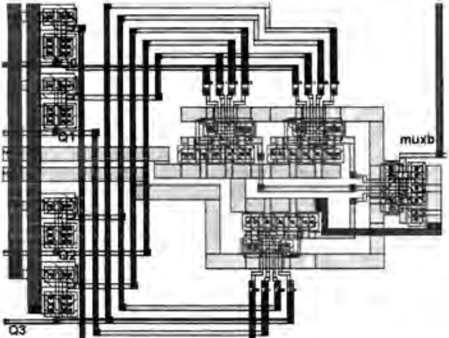

**Figure 43** muxb_fsm Cell Layout (MFSM).

see Appendix A, for implementation equation for circuit cfsmckt.

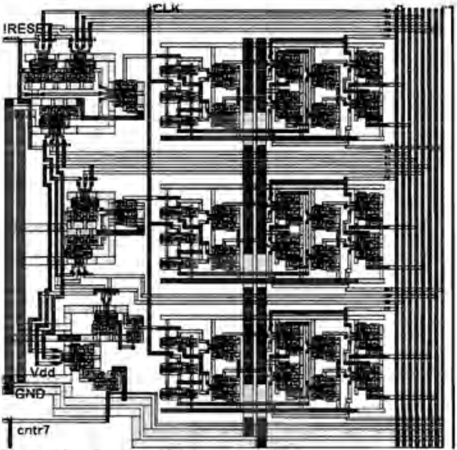

**Figure 44** cfsmckt Cell Layout (CFSM).

$ACK\_EN=\overline{(c\cdot rterm)+(e\cdot rterm)+(g\cdot rterm)+(\_\cdot rterm)}$
$rterm\cdot cntr7\cdot\overline{CLR}$

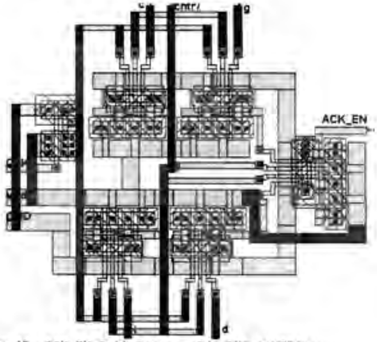

**Figure 45** ACK_EN Cell Layout (I/O BUS CONTROL).

$$LSSEN = A0 \cdot \overline{(I + ACLK) + (BCLK) + (DCLK)}$$

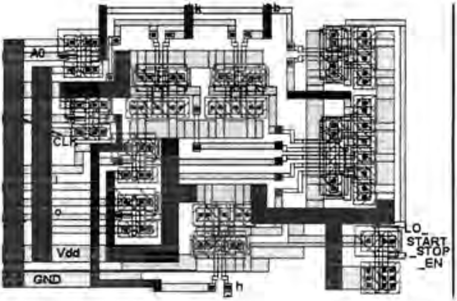

**Figure 46** LSSEN Cell Layout (I/O BUS CONTROL).

$HSSEN = \overline{BHE} \cdot (\overline{I} \cdot \overline{(D \cdot CLK)} \cdot \overline{(R \cdot CLK)} \cdot \overline{(O \cdot CLK)})$

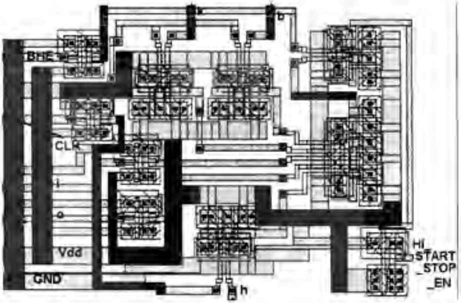

Figure 47 HSSEN Cell Layout (I/O BUS CONTROL).

COUNTER_EN=c+e+g+i+n
COUNTER_CLR=b+d+f+k+m

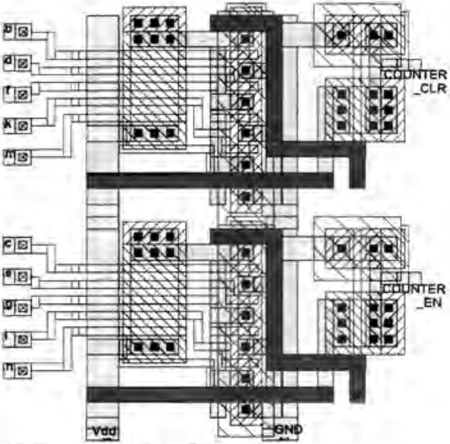

**Figure 48** cntr?con Cell Layout (MPEM).

OUT = IN1 · IN2

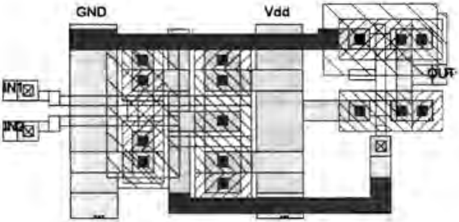

**Figure 49** Gate1 Cell Layout (General Gating Circuit)

$HI\_SR\_SHIFT\_CLK = \overline{BHE}\,(\overline{(\overline{CLK}\,(c*e*g*f)')\,(CLK\,(m+n))})$

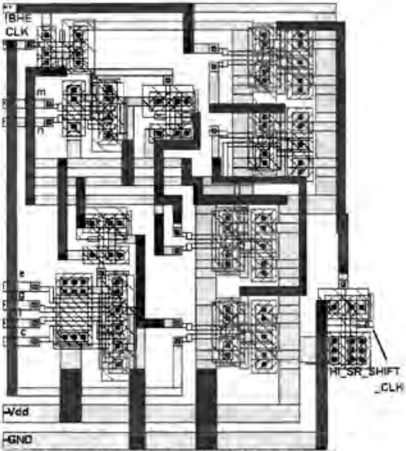

**Figure 50** Hisrshif Cell Layout (I/O BUS CONTROL).

$$LO\_SR\_SHIFT\_CLK=\overline{A0}\cdot(\overline{CLK\cdot(c+e+g+l)\cdot(CLK\cdot(m+n))})$$

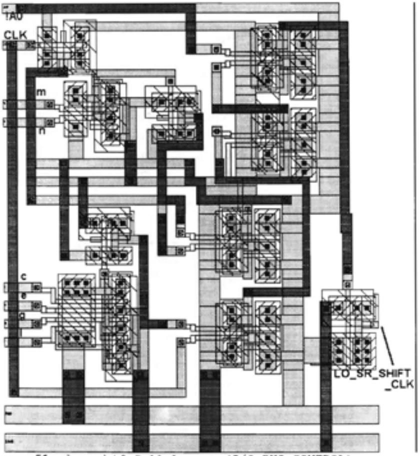

**Figure 51** losrshif Cell Layout (I/O BUS CONTROL).

$$HI\_SR\_LOAD\_CLK = \overline{BHE} \cdot \overline{CLK} \cdot (b + d + f + k)$$

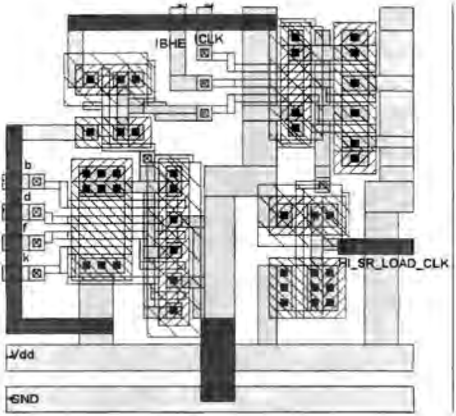

**Figure 52** hisrload Cell Layout (I/O BUS CONTROL).

$$LO\_CLK\_DISABLE = \overline{AO} + \overline{\overline{(a+i+o)} \cdot \overline{(h \cdot CLK)} \cdot \overline{(j \cdot CLK)} \cdot \overline{(n \cdot cntr7 \cdot CLK)}}$$

$$HI\_CLK\_DISABLE = \overline{BHE} \cdot \overline{\overline{(a+i+o)} \cdot \overline{(h \cdot CLK)} \cdot \overline{(j \cdot CLK)} \cdot \overline{(n \cdot cntr7 \cdot CLK)}}$$

$$STOP\_START\_SET = \overline{I} \cdot \overline{(n \cdot CLK)}$$

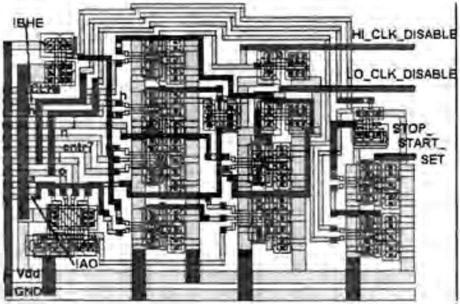

**Figure 53** loclkdis Cell Layout (I/O BUS CONTROL).

LO_SR_LOAD_CLK=R0·CLK·(b+d+f+k)

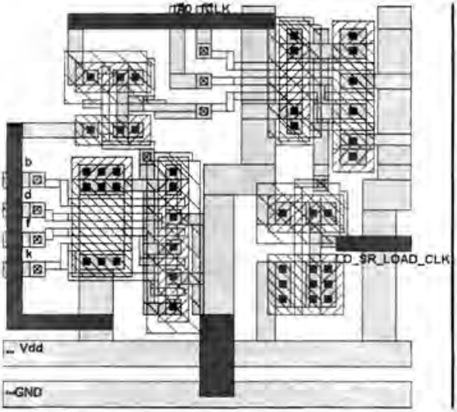

**Figure 54** losrload Cell Layout (I/O BUS CONTROL).

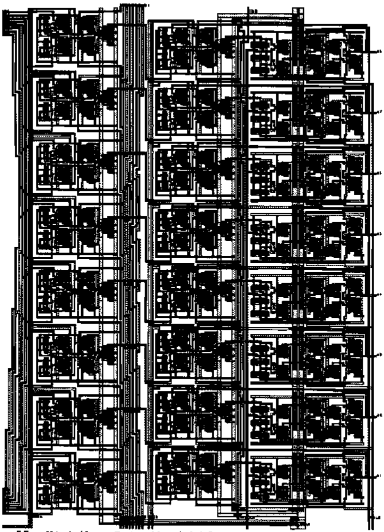

**Figure 55** High/Low Byte Mux/Shifter.

$LO\_SR\_BUS\_EN = \overline{RD}\cdot((lL(l\cdot cntr7\cdot CLK)) + ((dL(c\cdot cntr7\cdot CLK)) +$
$(eL(e\cdot cntr7\cdot \overline{CLK})) + (gL(g\cdot cntr7\cdot \overline{CLK}))(\overline{CLK}\cdot(b+d+f+k))$
$HI\_SR\_BUS\_EN = \overline{BHE}\cdot((lL(l\cdot cntr7\cdot \overline{CLK})) + ((dL(c\cdot cntr7\cdot \overline{CLK})) +$
$(eL(e\cdot cntr7\cdot \overline{CLK})) + (gL(g\cdot cntr7\cdot \overline{CLK}))(\overline{CLK}\cdot(b+d+f+k))$

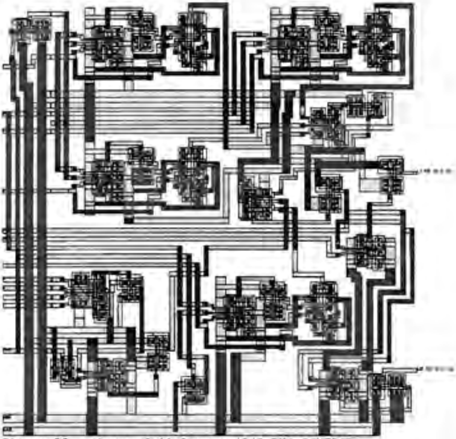

**Figure 56** srbusen Cell Layout (I/O BUS CONTROL).

# APPENDIX C: SIMULATION DATA

## A. MAIN FINITE STATE MACHINE SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 595 steps
initialization took 0 steps
> 010101010101010101010101010101010101010101010101:CLK
> 111111111111111111111111111111111111111111111111:PR
> 111111111111111111111111111111111111111111111111:CLR
> 111111111111111111111111111110000000000000000000:wr
> 111111100001111111111111111111111111100001111111:cntr7
> 111111111111111111110000111111111111111111100001:ack
> 000011111111111111111111111111111111111111111111:!ferro
> 111110000000000000000000000000110000000000000000:a
> 000001100000000000000000000000011000000000000000:b
> 000000011111100000000000000000001100000000000000:c
> 000000000000011000000000000000000000110000000000:d
> 000000000000000110000000000000000000111100000000:e
> 000000000000000000000000000000000000000001100000:f
> 000000000000000000000000000000000000000000110000:g
> 000000000000000000000000000000000000000000001100:h
> 000000000000000000000000000000000000000000000011:i
> 000000000000000011000000000000000000000000000000:j
> 000000000000000000110000000000000000000000000000:k
> 000000000000000000001100000000000000000000000000:l
> 000000000000000000000011110000000000000000000000:m
> 000000000000000000000000001100000000000000000000:n
> 000000000000000000000000000011000000000000000000:o
> 000000000000011001110000000000000011000000000000:muxa
> 000000000000000011100000000000000000000001100000:muxb
> 000001111111111111111111111100001111111111110000:s0
> 000001100000011001110000000000110011000011000000:s1
982 transistors, 505 nodes (0 pulled up)
```

## B. COUNTER FINITE STATE MACHINE SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 143 steps
initialization took 0 steps
>0101010101010101010101010101010101:CLK
>1111111111111111111011111111111111:CLR
>1111111111111111001111111111111111:PR
>1111111111111111111111111111111111:!RESET
>1000000000000110011000000000000110:cntr7
196 transistors, 104 nodes (0 pulled up)
```

## C. ACK_EN SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 28 steps
initialization took 0 steps
>010001000:c
>111110000:cntr7
>000000000:CLK
>000010001:e
>000100010:g
>001000100:j
>011110000:OUT
34 transistors, 25 nodes (0 pulled up)
```

## D. LSSEN SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 24 steps
initialization took 0 steps
>00000000001111111111:A0
>11000000001100000000:b
>00110000000011000000:i
>00001100000000110000:h
>00000011000000001100:k
>00000000110000000011:o
>01100110100110011010:CLK
>01111010110000000000:LO_START_STOP_EN
38 transistors, 27 nodes (0 pulled up)
```

## E. ESSEN SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 28 steps
initialization took 0 steps
> 0000000000000000000000000000000000000000000000000000000000000000:CLK
> 1111111111111111111111111111111111111111111111111111111111111111:!BHE
> 0101010101010101010101010101010101010101010101010101010101010101:c
> 0011001100110011001100110011001100110011001100110011001100110011:e
> 0000000000000000111111111111111100000000000000001111111111111111:g
> 0000000000000000000000000000000011111111111111111111111111111111:l
> 0000111100001111000011110000111100001111000011110000111100001111:m
> 0000000011111111000000001111111100000000111111110000000011111111:n
> 0000111111111111000011111111111100001111111111110000111111111111:HI_SR_SH
IFT_CLK
36 transistors, 28 nodes (0 pulled up)
```

```
ESIM (V3.5 03/27/91)
initialization took 28 steps
initialization took 0 steps
> 1111111111111111111111111111111111111111111111111111111111111111:CLK
> 0000000000000000000000000000000000000000000000000000000000000000:!BHE
> 0101010101010101010101010101010101010101010101010101010101010101:c
> 0011001100110011001100110011001100110011001100110011001100110011:e
> 0000000000000000111111111111111100000000000000001111111111111111:g
> 0000000000000000000000000000000011111111111111111111111111111111:l
> 0000111100001111000011110000111100001111000011110000111100001111:m
> 0000000011111111000000001111111100000000111111110000000011111111:n
> 0000000000000000000000000000000000000000000000000000000000000000:HI_SR_SH
IFT_CLK
36 transistors, 28 nodes (0 pulled up)
```

## F. BISRSHIF SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 28 steps
initialization took 0 steps
> 1111111111111111111111111111111111111111111111111111111111111111:CLK
> 1111111111111111111111111111111111111111111111111111111111111111:!BHE
> 0101010101010101010101010101010101010101010101010101010101010101:c
> 0011001100110011001100110011001100110011001100110011001100110011:e
> 0000000000000000111111111111111100000000000000001111111111111111:g
> 0000000000000000000000000000000011111111111111111111111111111111:l
> 0000111100001111000011110000111100001111000011110000111100001111:m
> 0000000011111111000000001111111100000000111111110000000011111111:n
> 0111011101110111111111111111111111111111111111111111111111111111:HI_SR_SH
IFT_CLK
36 transistors, 28 nodes (0 pulled up)
```

```
ESIM (V3.5 03/27/91)
initialization took 28 steps
initialization took 0 steps
> 0000000000000000000000000000000000000000000000000000000000000000:CLK
> 1111111111111111111111111111111111111111111111111111111111111111:!BHE
> 0101010101010101010101010101010101010101010101010101010101010101:c
> 0011001100110011001100110011001100110011001100110011001100110011:e
> 0000000000000000111111111111111100000000000000001111111111111111:g
> 0000000000000000000000000000000011111111111111111111111111111111:l
> 0000111100001111000011110000111100001111000011110000111100001111:m
> 0000000011111111000000001111111100000000111111110000000011111111:n
> 0000111111111111000011111111111100001111111111110000111111111111:HI_SR_SH
IFT_CLK
36 transistors, 28 nodes (0 pulled up)
```

```
ESIM (V3.5 03/27/91)
initialization took 28 steps
initialization took 0 steps
> 1111111111111111111111111111111111111111111111111111111111111111:CLK
> 0000000000000000000000000000000000000000000000000000000000000000:!BHE
> 0101010101010101010101010101010101010101010101010101010101010101:c
> 0011001100110011001100110011001100110011001100110011001100110011:e
> 0000000000000000111111111111111100000000000000001111111111111111:g
> 0000000000000000000000000000000011111111111111111111111111111111:l
> 0000111100001111000011110000111100001111000011110000111100001111:m
> 0000000011111111000000001111111100000000111111110000000011111111:n
> 0000000000000000000000000000000000000000000000000000000000000000:HI_SR_SH
IFT_CLK
38 transistors, 28 nodes (0 pulled up)
```

```
ESIM (V3.5 03/27/91)
initialization took 28 steps
initialization took 0 steps
> 0000000000000000000000000000000000000000000000000000000000000000:CLK
> 0000000000000000000000000000000000000000000000000000000000000000:!BHE
> 0101010101010101010101010101010101010101010101010101010101010101:c
> 0011001100110011001100110011001100110011001100110011001100110011:e
> 0000000000000000111111111111111100000000000000001111111111111111:g
> 0000000000000000000000000000000011111111111111111111111111111111:l
> 0000111100001111000011110000111100001111000011110000111100001111:m
> 0000000011111111000000001111111100000000111111110000000011111111:n
> 0000000000000000000000000000000000000000000000000000000000000000:HI_SR_SH
IFT_CLK
38 transistors, 28 nodes (0 pulled up)
```

G. LOSRSHIF SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 28 steps
initialization took 0 steps
> 11111111111111111111111111111111111111111111111111111111111111111:CLK
> 00000000000000000000000000000000000000000000000000000000000000000:!A0
> 01010101010101010101010101010101010101010101010101010101010101010:c
> 00110011001100110011001100110011001100110011001100110011001100110:e
> 00001111000011110000111100001111000011110000111100001111000011110:g
> 00000000111111110000000011111111000000001111111100000000111111110:l
> 00000000000000001111111111111111000000000000000011111111111111111:n
> 00000000000000000000000000000000111111111111111111111111111111111:m
> 00000000000000000000000000000000000000000000000000000000000000000:LO_SR_S
HIFT_CLK
36 transistors, 28 nodes (0 pulled up)

ESIM (V3.5 03/27/91)
initialization took 28 steps
initialization took 0 steps
> 11111111111111111111111111111111111111111111111111111111111111111:CLK
> 11111111111111111111111111111111111111111111111111111111111111111:!A0
> 01010101010101010101010101010101010101010101010101010101010101010:c
> 00110011001100110011001100110011001100110011001100110011001100110:e
> 00001111000011110000111100001111000011110000111100001111000011110:g
> 00000000111111110000000011111111000000001111111100000000111111110:l
> 00000000000000001111111111111111000000000000000011111111111111111:n
> 00000000000000000000000000000000111111111111111111111111111111111:m
> 01111111111111110111111111111111011111111111111101111111111111110:LO_SR_S
HIFT_CLK
36 transistors, 28 nodes (0 pulled up)
```

## H. HISRLOAD SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 15 steps
initialization took 0 steps
> 0101010101010101010101010101010101010101010101010101010101010101:!CLK
> 0011001100110011001100110011001100110011001100110011001100110011:!BHE
> 0000111100001111000011110000111100001111000011110000111100001111:b
> 0000000011111111000000001111111100000000111111110000000011111111:d
> 0000000000000000111111111111111100000000000000001111111111111111:f
> 0000000000000000000000000000000011111111111111111111111111111111:k
> 0000000100010001000100010001000100010001000100010001000100010001:HI_SR_LO
AD_CLK
18 transistors, 17 nodes (0 pulled up)
```

## I. LOSRLOAD SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 15 steps
initialization took 0 steps
> 0101010101010101010101010101010101010101010101010101010101010101:!A0
> 0011001100110011001100110011001100110011001100110011001100110011:!CLK
> 0000111100001111000011110000111100001111000011110000111100001111:b
> 0000000011111111000000001111111100000000111111110000000011111111:d
> 0000000000000000111111111111111100000000000000001111111111111111:f
> 0000000000000000000000000000000011111111111111111111111111111111:k
> 0000000100010001000100010001000100010001000100010001000100010001:LO_SR_LO
AD_CLK
18 transistors, 17 nodes (0 pulled up)
```

## J. LOCLKDIS SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 41 steps
initialization took 0 steps
> 1110000000000011100000000000:n
> 1000000000000010000000000000:cntr7
> 0001100000000000011000000000:a
> 0000011000000000000110000000:i
> 0000000110000000000000110000:h
> 0000000001100000000000001100:j
> 0000000000001100000000000011:o
> 1010101010101101010101010101:CLK
> 1111111111111100000000000000:!A0
> 0000000000000011111111111111:!BHE
> 0000000000000000001111101011:HI_CLK_DISABLE
> 0001111101011000000000000000:LO_CLK_DISABLE
> 0000011000001000000110000010:STOP_START_SET
54 transistors, 39 nodes (0 pulled up)
```

## K. HIGH/LOW BYTE MUX/SHIFTER SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 740 steps
initialization took 0 steps
> 0101010101010101010101010101010101010101010101010101010101010101:CLK
> 1111111111111111111111111111111111111111111111111111111111111111:PR
> 1111111111111111111111111111111111111111111111111111111111111111:CLR
> 0000000000000000000000000000000000000000000000000000000000000000:SINL
> 0000000000000000000000000000000000000000000000000000000000000000:SINR
> 0000000000000000110000000000000000000000000000001100000000000000:muxa
> 0000000000000000000000000000000011000000000000001100000000000000:muxb
> 1100000000000000110000000000000011000000000000001100000000000000:S1
> 1111111111111111111111111111111111111111111111111111111111111111:S0
> 0000000000000000000000000000000000000000000000000000000000000000:b01
> 1111111111111111111111111111111111111111111111111111111111111111:b02
> 0000000000000000000000000000000000000000000000000000000000000000:b03
> 1111111111111111111111111111111111111111111111111111111111111111:b04
> 1111111111111111111111111111111111111111111111111111111111111111:b05
> 0000000000000000000000000000000000000000000000000000000000000000:b06
> 1111111111111111111111111111111111111111111111111111111111111111:b07
> 1111111111111111111111111111111111111111111111111111111111111111:b08
> 1111111111111111111111111111111111111111111111111111111111111111:b11
> 0000000000000000000000000000000000000000000000000000000000000000:b12
> 1111111111111111111111111111111111111111111111111111111111111111:b13
> 1111111111111111111111111111111111111111111111111111111111111111:b14
> 1111111111111111111111111111111111111111111111111111111111111111:b15
> 1111111111111111111111111111111111111111111111111111111111111111:b16
> 0000000000000000000000000000000000000000000000000000000000000000:b17
> 1111111111111111111111111111111111111111111111111111111111111111:b18
> 1111111111111111111111111111111111111111111111111111111111111111:b21
> 1111111111111111111111111111111111111111111111111111111111111111:b22
> 1111111111111111111111111111111111111111111111111111111111111111:b23
> 1111111111111111111111111111111111111111111111111111111111111111:b24
> 0000000000000000000000000000000000000000000000000000000000000000:b25
> 1111111111111111111111111111111111111111111111111111111111111111:b26
> 1111111111111111111111111111111111111111111111111111111111111111:b27
> 1111111111111111111111111111111111111111111111111111111111111111:b28
> 1111111111111111111111111111111111111111111111111111111111111111:b31
> 1111111111111111111111111111111111111111111111111111111111111111:b32
> 1111111111111111111111111111111111111111111111111111111111111111:b33
> 0000000000000000000000000000000000000000000000000000000000000000:b34
> 1111111111111111111111111111111111111111111111111111111111111111:b35
> 1111111111111111111111111111111111111111111111111111111111111111:b36
> 1111111111111111111111111111111111111111111111111111111111111111:b37
> 1111111111111111111111111111111111111111111111111111111111111111:b38
> 0001100111100111111001111111100111111111100111111111111001111111:A1
> 0110011110011110000111111110011001111110011111110011100111111110:A2
> 0001111001111100001111111100110000111001111110000110011111111000:A3
> 0111100111100000011111001000000100111110000000000011111111100000:A4
> 0110011110000000111001100000000011111100000000001111111110000000:A5
> 0001110000000000110011000000000011111100000000001111110000000000:A6
> 0111100000000000110000000000000011110000000000001111000000000000:A7
> 0110000000000000110000000000000011000000000000001100000000000000:A8
944 transistors, 515 nodes (0 pulled up)
```

## L. SRBUSEN SIMULATION

```
ESIM (V3.5 03/27/91)
initialization took 103 steps
initialization took 0 steps
> 11111111111111111111000000000000000000:!A0
> 10000000000000000000111111111111111111:!BHE
> 00011000000000000000000100000000000000:cntr7
> 01111000000000000000011100000000000000:c
> 00000110000000000000000110000000000000:e
> 00000001100000000000000001100000000000:g
> 00000000011000000000000000011000000000:l
> 10110101010000000111110111010100000001111:CLK
> 00000000000001000100000000000000010001000:b
> 00000000000001000100000000000000001000100:d
> 00000000000000100010000000000000000100010:f
> 00000000000000100010000000000000000010001:k
> 01110111111011110000000000000000000000000:LD_SR_BUS_EN
> 00000000000000000000011111111011110000:HI_SR_BUS_EN
114 transistors, 67 nodes (0 pulled up)
```

## N. PAD-TO-PAD VERIFICATION

### 1. Read Cycle Simulation

```
[illegible]
initialization took 9982 steps
initialization took 7793 steps
initialization took 3949 steps
initialization took 768 steps
initialization took 69 steps
initialization took 69 steps
initialization took 8 steps
initialization took 8 steps
> [illegible] CLK
> [illegible] ENAB
> [illegible] WR
> [illegible] RD
> [illegible] ADV
> [illegible] SCAL
> [illegible] SCAR
> [illegible] SCLL
> [illegible] SCLR
> [illegible] a
> [illegible] b
> [illegible] c
> [illegible] d
> [illegible] e
> [illegible] f
> [illegible] g
> [illegible] h
> [illegible] i
> [illegible] j
> [illegible] k
> [illegible] l
> [illegible] m
> [illegible] n
> [illegible] o
> [illegible] D0
> [illegible] D1
> [illegible] D2
> [illegible] D3
> [illegible] D4
> [illegible] D5
> [illegible] D6
> [illegible] D7
> [illegible] D8
> [illegible] D9
> [illegible] D10
> [illegible] D11
> [illegible] D12
> [illegible] D13
> [illegible] D14
> [illegible] D15
> [illegible] DHE
> [illegible] A0
> [illegible] A1
> [illegible] A2
> [illegible] A3
> [illegible] A4
> [illegible] A5
```

```
>11111111111111111111111111111111111111111111111111111111111111111111111111111111111:A6
>11111111111111111111111111111111111111111111111111111111111111111111111111111111111:A7
>11111111111111111111111111111111111111111111111111111111111111111111111111111111111:A8
>11111111111111111111111111111111111111111111111111111111111111111111111111111111111:A9
>11111111111111111111111111111111111111111111111111111111111111111111111111111111111:A10
>11111111111111111111111111111111111111111111111111111111111111111111111111111111111:A11
>00000000011111111111111110011111111111111100000111111111111111000000000000000000000:$DAH_IN
>00000000011111111111111110011111111111111100000111111111111111000000000000000000000:$DAL_IN
>11111111100000000000000001100000000000000000111000000000000000011111111111111111111:SDAH_OUT
>11111111100000000000000001100000000000000000111000000000000000011111111111111111111:SDAL_OUT
>11111111111111111111111111111111111111111111111111111111111111111111111111111111100:LD_DATA_OUT
>11111111111111111111111111111111111111111111111111111111111111111111111111111111100:HI_DATA_OUT
4720 transistors, 2172 nodes (42 pulled up)
```

## 2. Write Cycle Simulation

```
ESIM (V3.5 03/27/91)
initialization took 8978 steps
initialization took 2682 steps
initialization took 3440 steps
initialization took 764 steps
initialization took 68 steps
initialization took 49 steps
initialization took 4 steps
initialization took 0 steps
>10101010101010101010101010101010101010101010101010101010101010101010101010:CLK
>11111100000000000000000000000000000000000000000000000000000000000000000000:FERRD
>00000000000000000000000000000000000000000000000000000000000000000000000000:WR
>11111111111111111111111111111111111111111111111111111111111111111111111111:RD
>11111100000000000000000000000000000000000000000000000000000000000000000011:RDY
>00000000100110011111100000011111110011111001111100000111111111001111000000:$DAL
>00000000100110011111100000011111110011111001111100000111111111001111000000:$DAH
>11111111010101010101010101010101010101010101010101010101010101010101010111:SCLL
>11111111010101010101010101010101010101010101010101010101010101010101010111:SCLH
>11111111000000000000000000000000000000000000000000000000000000000000000000:a
>00000000011000000000000000000000000000000000000000000000000000000000000000:b
>00000000000111111111111111110000000000000000000000000000000000000000000000:c
>00000000000000000000000000110000000000000000000000000000000000000000000000:d
>00000000000000000000000000011111111111111111111110000000000000000000000000:e
>00000000000000000000000000000000000000000000000001100000000000000000000000:f
>00000000000000000000000000000000000000000000000000011111111111111111110000:g
>00000000000000000000000000000000000000000000000000000000000000000000001100:h
>00000000000000000000000000000000000000000000000000000000000000000000000011:i
>00000000000000000000000000000000000000000000000000000000000000000000000000:j
>00000000000000000000000000000000000000000000000000000000000000000000000000:k
>00000000000000000000000000000000000000000000000000000000000000000000000000:l
>00000000000000000000000000000000000000000000000000000000000000000000000000:m
>00000000000000000000000000000000000000000000000000000000000000000000000000:n
>00000000000000000000000000000000000000000000000000000000000000000000000000:o
>11111111111111111111111111111111111111111111111111111111111111111111111111:D0
>11111111111111111111111111111111111111111111111111111111111111111111111111:D1
>00000000000000000000000000000000000000000000000000000000000000000000000000:D2
>11111111111111111111111111111111111111111111111111111111111111111111111111:D3
>11111111111111111111111111111111111111111111111111111111111111111111111111:D4
>11111111111111111111111111111111111111111111111111111111111111111111111111:D5
>11111111111111111111111111111111111111111111111111111111111111111111111111:D6
>11111111111111111111111111111111111111111111111111111111111111111111111111:D7
>11111111111111111111111111111111111111111111111111111111111111111111111111:D8
>11111111111111111111111111111111111111111111111111111111111111111111111111:D9
>00000000000000000000000000000000000000000000000000000000000000000000000000:D10
```

```
> 11111111111111111111111111111111111111111111111111111111111111111111:D11
> 11111111111111111111111111111111111111111111111111111111111111111111:D12
> 11111111111111111111111111111111111111111111111111111111111111111111:D13
> 11111111111111111111111111111111111111111111111111111111111111111111:D14
> 11111111111111111111111111111111111111111111111111111111111111111111:D15
> 00000000000000000000000000000000000000000000000000000000000000000000:BHE
> 00000000000000000000000000000000000000000000000000000000000000000000:A0
> 11111111111111111111111111111111111111111111111111111111111111111111:A1
> 11111111111111111111111111111111111111111111111111111111111111111111:A2
> 00000000000000000000000000000000000000000000000000000000000000000000:A3
> 11111111111111111111111111111111111111111111111111111111111111111111:A4
> 11111111111111111111111111111111111111111111111111111111111111111111:A5
> 11111111111111111111111111111111111111111111111111111111111111111111:A6
> 11111111111111111111111111111111111111111111111111111111111111111111:A7
> 11111111111111111111111111111111111111111111111111111111111111111111:A8
> 11111111111111111111111111111111111111111111111111111111111111111111:A9
> 11111111111111111111111111111111111111111111111111111111111111111111:A10
> 11111111111111111111111111111111111111111111111111111111111111111111:A11
> 11111111100000000000000001100000000000000000110000000000000000011111:SDAH_OUT
> 11111111100000000000000001100000000000000000110000000000000000011111:SDAL_OUT
> 00000000011111111111111110011111111111111111001111111111111111100000:SDAH_IN
> 00000000011111111111111110011111111111111111001111111111111111100000:SDAL_IN
> 11111111111111111111111111111111111111111111111111111111111111111111:HI_DATA_OUT
> 11111111111111111111111111111111111111111111111111111111111111111111:LO_DATA_OUT
4720 transistors, 2172 nodes (42 pulled up)
```

APPENDIX D: CIRCUIT SCHEMATICS

## A. MICROINTERFACE WITHOUT LINE DRIVERS

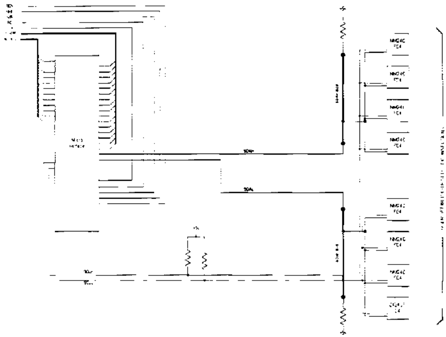

**Figure 57** Microinterface Without Line Drivers.

## B. MICROINTERFACE WITH SDA BUS LINE DRIVERS

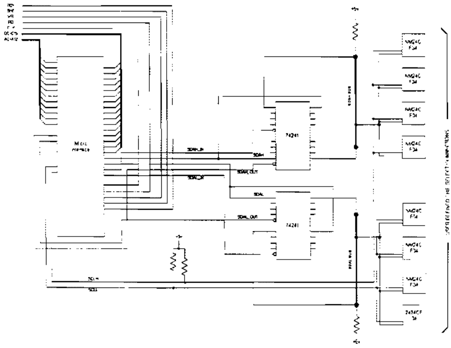

**Figure 58** Microinterface With Line Drivers.

## APPENDIX E: PRELIMINARY DATA SHEET

Component Name: Microinterface Chip

General Description:

Microinterface is an interface between National Semiconductor's NM24CF04, nonvolatile, serial-access, ferroelectric memory device, and Intel's 8086 microprocessor. A pin diagram is shown in Figure 59.

Pin description:

| | |
|---|---|
| A0-A11 | Address input pins |
| D0-D15 | Bidirectional parallel data bus pins |
| SDAH/L | Bidirectional serial data bus pins |
| RD/WR | Read/Write request input (active low) |
| BHE | Bus High Enable (active low) |
| FERRO | Chip select input (active low) |
| CLK | 100 KHz clock signal input |
| SDAH/L_OUT, SDAH/L_IN, HI/LO_DATA_OUT | External driver enables (active low) |
| SCLH/L | Serial Clock outputs to NM24CF04 chips |
| RDY | READY output (active low) |
| RESET | Input to reset Main Finite State Machine |

Static Power Dissipation (no input): 821 μW
Dynamic Power Dissipation (worst case) : 1.64 W

**Figure 59** Preliminary Pinout.

## LIST OF REFERENCES

1. T.C. Gonter, "A Microprocessor Interface for the NM24CF04 Serial-Access Ferroelectric Memory," *Masters Thesis*, Naval Postgraduate School, Monterey, CA, Dec 1991.

2. D.A. Carver, "A Ferroelectric Material Development Test Chip," *Proceedings of The Second Symposium on Integrated Ferroelectrics*, pp. 125-134, Mar 1990.

3. R.L. Wiker, "Ferroelectric Memories for Military Applications," *Proceedings of the Second Symposium on Integrated Ferroelectrics*, pp. 163-174, Monterey, CA, Mar 1990.

4. National Semiconductor, *NM24CF04, 4096-Bit (512x8) CMOS Serial Nonvolatile Memory*, November 1990.

5. Intel Corporation, *iAPX 86/88, 186/188 User's Manual Hardware Reference*, 1985.

6. Computer Science Division, University of California at Berkeley, *Berkeley Cad Tools User's Manual*, University of California at Berkeley, 1986.

7. Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, *SPICE3C1 User's Guide*, by T. Quarles, and others 27 April 1987.

8. Weste, N. H., and Eshraghian, K., *Principles of CMOS VLSI Design*, Addison-Wesley Publishing Co., 1988.

## INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria VA 22304-6145 | 2 |
| 2. | Library, Code 52<br>Naval Postgraduate School<br>Monterey CA 93943-5101 | 2 |
| 3. | Department Chairman, Code EC<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey CA 93943-5121 | 1 |
| 4. | Professor Douglas Fouts, Code EC/Fs<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey CA 93943-5121 | 2 |
| 5. | Professor Chi-Hwa Lee, Code EC/Le<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey CA 93943-5121 | 1 |
| 6. | LT James H. Dickerson<br>Portsmouth Naval Shipyard<br>Portsmouth NH 03804 | 2 |